

LAPORAN TUGAS BESAR II

IF4070

Pengembangan Graf Pengetahuan dan Sistem RAG untuk Domain Dota 2



oleh

SA'AD ABDUL HAKIM - 13522092

RAYHAN FADHLAN AZKA - 13522095

RAYENDRA ALTHAF TARAKA NOOR - 13522107

**PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO & INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG**

Desember 2025

Daftar Isi

I. Graf pengetahuan.....	3
I.1. Pemanfaatan.....	3
I.2. Deskripsi Label dan Simpul.....	3
I.3. Keterkaitan dengan Ontologi.....	5
II. Sistem RAG.....	5
II.1. Fungsionalitas Penting.....	5
II.2. Pemanfaatan Graf Pengetahuan.....	6
III. Pembahasan.....	7
III.1. Hasil dari Sistem RAG.....	7
III.2. Perbandingan antara Sistem RAG dan Sistem Berbasis Pengetahuan.....	13
III.3. Keterbatasan Graf Pengetahuan dan RAG.....	14
IV. Kesimpulan.....	16
Lampiran.....	18
Matriks Kontribusi.....	18

I. Graf pengetahuan

I.1. Pemanfaatan

Graf pengetahuan dibangun menggunakan Neo4j untuk merepresentasikan data kompleks dari permainan Dota 2, memodelkan hubungan semantik antara Hero, Ability, Item, dan atribut pendukung lainnya secara terstruktur. Dalam arsitektur sistem, graf ini berfungsi vital sebagai memori faktual atau *ground truth* yang memungkinkan agen cerdas (LLM) melakukan kueri relasional yang mendalam, seperti menelusuri dependensi resep item yang bertingkat atau mencari hero berdasarkan kombinasi atribut spesifik, kemampuan navigasi yang sulit dicapai jika data hanya disimpan dalam format teks datar atau tabular. Selain menjamin akurasi data mekanik, struktur ini juga berperan krusial sebagai target penerjemahan modul *Text-to-Cypher*, memastikan sistem memberikan jawaban yang presisi dan deterministik berdasarkan fakta yang tersimpan.

I.2. Deskripsi Label dan Simpul

Berdasarkan implementasi kode dan definisi skema, graf pengetahuan ini dimodelkan menggunakan struktur Nodes (Simpul) dan Relationships (Relasi) berikut:

A. Simpul (Nodes)

Entitas utama dalam permainan direpresentasikan sebagai **Nodes** dengan properti spesifik:

1. **Hero**: Merepresentasikan karakter yang dimainkan.
Properties: name, internalId, baseHealth, baseMana, moveSpeed, legs, attackRange.
2. **Ability**: Merepresentasikan kemampuan aktif atau pasif yang dimiliki oleh **Hero**.
Properties: name, key, description, manaCost, cooldown.
3. **Item**: Merepresentasikan barang atau peralatan yang dapat dibeli dan digunakan dalam permainan.
Properties: name, key, cost, lore, notes, attributes.
4. **ItemAbility**: Entitas khusus yang memodelkan kemampuan aktif/pasif yang melekat pada sebuah **Item** (misalnya efek "Active" dari *Black King Bar*).
Properties: name, description, type.
5. **Facet**: Fitur mekanik baru Dota 2 yang merepresentasikan variasi gaya bermain atau *playstyle* dari **Hero**.
Properties: name, description.

Selain entitas dinamis di atas, terdapat simpul kategori (Static/Enumeration Nodes) untuk normalisasi data:

1. **Attribute**: Atribut utama **Hero** (contoh: "Strength", "Agility", "Intelligence", "Universal").

2. **Role:** Peran strategis **Hero** dalam tim (contoh: "Carry", "Support", "Nuker", "Disabler").
3. **AttackType:** Tipe serangan **Hero** (contoh: "Melee", "Ranged").
4. **DamageType:** Jenis kerusakan yang dihasilkan oleh **Ability** (contoh: "Magical", "Physical", "Pure").
5. **Behavior:** Sifat teknis atau perilaku dari target **Ability** atau **Item** (contoh: "Passive", "Unit Target", "Channeled").

B. Sisi/Relasi (Relationships)

Hubungan semantik antar simpul didefinisikan melalui Relationships berikut, yang membentuk ontologi graf:

- Karakteristik Hero:
 - (:Hero)-[:HAS_PRIMARY_ATTR]->(:Attribute): Menghubungkan **Hero** dengan atribut utamanya.
 - (:Hero)-[:HAS_ROLE]->(:Role): Mengaitkan **Hero** dengan peran strategisnya.
 - (:Hero)-[:HAS_ATTACK_TYPE]->(:AttackType): Menentukan apakah serangan **Hero** bertipe *Melee* atau *Ranged*.
 - (:Hero)-[:HAS_FACET]->(:Facet): Menghubungkan **Hero** dengan opsi **Facet** yang dimilikinya.
- Kemampuan (Abilities):
 - (:Hero)-[:HAS_ABILITY]->(:Ability): Menghubungkan **Hero** dengan daftar **Ability** miliknya.
 - (:Hero)-[:HAS_ULTIMATE]->(:Ability): Relasi spesifik untuk menandai kemampuan pamungkas (*ultimate*) dari **Hero** (biasanya indeks ke-5).
 - (:Ability)-[:DEALS_DAMAGE_TYPE]->(:DamageType): Menentukan jenis *damage* yang dihasilkan oleh **Ability**.
 - (:Ability)-[:HAS_BEHAVIOR]->(:Behavior): Menentukan mekanisme atau perilaku eksekusi **Ability**.
- Mekanisme Item:
 - (:Item)-[:REQUIRES_COMPONENT]->(:Item): Memodelkan struktur resep, di mana satu **Item** membutuhkan **Item** lain sebagai bahan pembuatannya (bersifat hierarkis).
 - (:Item)-[:HAS_ABILITY]->(:ItemAbility): Menghubungkan **Item** dengan efek aktif atau pasif uniknya.
 - (:Item)-[:HAS_BEHAVIOR]->(:Behavior): Menjelaskan sifat penggunaan **Item** (misalnya *Instant* atau *Target Unit*).

I.3. Keterkaitan dengan Ontologi

Graf pengetahuan ini berfungsi sebagai instansiasi praktis dari sebuah ontologi domain permainan Dota 2. Lebih dari sekadar penyimpanan data, graf ini mengimplementasikan struktur semantik yang memetakan bagaimana dunia Dota 2 bekerja. Struktur ontologi ini dapat diuraikan ke dalam tiga komponen utama:

- a. **Konsep (Concepts):** Direpresentasikan oleh **Node Labels**. Dalam kerangka ontologi, label ini berfungsi sebagai **T-Box** (*Terminological Box*) yang mendefinisikan kelas-kelas atau kategori entitas yang valid dalam domain pengetahuan ini. Sebagai contoh, label seperti Hero, Ability, dan Item bukan sekadar nama tabel, melainkan definisi abstrak dari entitas yang eksis dalam semesta Dota 2.
- b. **Relasi (Roles/Properties):** Direpresentasikan oleh **Edges** (sisi) yang menghubungkan antar simpul. Relasi ini memberikan makna semantik yang mendalam pada struktur graf:
 - **Composition:** Relasi seperti `(:Item)-[:REQUIRES_COMPONENT]->(:Item)` secara semantik berarti "Item A terdiri dari (atau membutuhkan) Item B". Ini memodelkan hierarki resep yang bersifat rekursif.
 - **Attribution:** Relasi yang mendefinisikan karakteristik yang melekat pada sebuah konsep. Contohnya, `(:Hero)-[:HAS_ROLE]->(:Role)` atau `(:Ability)-[:DEALS_DAMAGE_TYPE]->(:DamageType)` menjelaskan sifat atau properti inheren dari entitas tersebut, bukan sekadar hubungan kepemilikan biasa.
- c. **Asersi (Assertions):** Direpresentasikan oleh data instansiasi (Instance Data). Data spesifik yang diinjeksi dari file JSON (seperti `MERGE (h:Hero {name: "Anti-Mage"})`) berfungsi sebagai **A-Box** (*Assertion Box*) yang mengisi kerangka ontologi (T-Box). Keberadaan asersi faktual ini memungkinkan **penalaran deduktif** (*reasoning*); sebagai contoh, sistem dapat menyimpulkan dependensi transitif di mana kepemilikan item tingkat lanjut (seperti *Butterfly*) secara otomatis mengimplikasikan kebutuhan terhadap komponen penyusunnya (*Eaglesong*), meskipun hubungan langsung antara Hero dan komponen tersebut tidak didefinisikan secara eksplisit.

II. Sistem RAG

II.1. Fungsionalitas Penting

Sistem ini menerapkan arsitektur *Graph Retrieval-Augmented Generation* (Graph RAG) yang bertumpu pada model bahasa besar (LLM) **Llama 4 Scout**. Pemilihan model ini didasarkan pada kemampuannya yang mumpuni dalam logika pemrograman (penting untuk sintaks Cypher) serta pemrosesan bahasa alami, yang diakses melalui API dari <https://freeinference.org>. Secara fungsional, sistem beroperasi melalui lima tahapan linear yang dikoordinasikan oleh skrip utama (`main.py`), yaitu:

1. **Input Pengguna (User Interface)** Interaksi dimulai melalui antarmuka baris perintah (CLI). Pengguna memasukkan pertanyaan dalam bahasa alami (misalnya: *"What items does Anti-Mage need?"*). Sistem juga dilengkapi mekanisme penanganan interupsi (*KeyboardInterrupt*) untuk pengalaman pengguna yang lebih responsif.
2. **Text-to-Cypher Translation** Ini adalah inti dari kecerdasan sistem. Modul *TextToCypher* menerima pertanyaan pengguna dan skema graf (*schema_example.txt*) sebagai konteks. Dengan menggunakan *system prompt* khusus yang menginstruksikan peran sebagai "ahli Neo4j", LLM menerjemahkan intensi pengguna menjadi kueri Cypher yang valid secara sintaksis. Modul ini juga memiliki fitur validasi semantik: jika pertanyaan dinilai tidak relevan dengan domain Dota 2, sistem akan mengembalikan token *IRRELEVANT* untuk mencegah halusinasi.
3. **Eksekusi Graf (Graph Execution)** Kueri Cypher yang dihasilkan kemudian dieksekusi oleh *GraphDatabaseDriver* ke basis data Neo4j. Proses ini memastikan bahwa data yang diambil adalah data faktual yang tersimpan dalam graf, bukan pengetahuan implisit dari LLM. Jika terjadi kesalahan sintaks pada kueri yang dibuat LLM, blok penanganan *error* (*try-except ClientError*) akan menangkapnya agar sistem tidak *crash*.
4. **Augmentasi Konteks (Context Augmentation)** Hasil mentah dari database yang dapat berupa daftar simpul, properti, atau relasi diambil dan diserialisasi menjadi format string teks. Langkah ini menjembatani format data terstruktur dari database graf agar dapat "dibaca" dan dipahami kembali oleh LLM pada tahap selanjutnya.
5. **Pembangkitan Respons (Response Generation)** Tahap akhir ditangani oleh modul *ResponseGenerator*. Modul ini menerima tiga *input* sekaligus: (1) pertanyaan asli pengguna, (2) kueri Cypher yang digunakan (untuk konteks logika), dan (3) hasil data mentah dari Neo4j (sebagai fakta). LLM kemudian merangkai jawaban akhir dalam bahasa alami yang koheren, akurat, dan langsung menjawab inti pertanyaan berdasarkan fakta yang ditemukan.

II.2. Pemanfaatan Graf Pengetahuan

Sistem ini mengadopsi pendekatan **Graph RAG** yang berbeda secara fundamental dari arsitektur RAG konvensional. Berbeda dengan metode umum yang mengandalkan pemotongan teks (*text chunking*) dan pencarian vektor (*vector search*) pada dokumen korpus, sistem ini memanfaatkan struktur topologi graf untuk memahami konteks data secara presisi, seperti hierarki pohon resep item atau hubungan spesifik antara *hero* dan *skill*-nya. Pendekatan ini memberikan implikasi signifikan pada kualitas konteks yang diproses:

1. **Konteks Terstruktur:** Informasi yang disuplai ke LLM bukan berupa potongan paragraf teks acak yang diambil berdasarkan kemiripan semantik semata, melainkan berupa fakta terstruktur yang diekstraksi langsung dari logika graf.
2. **Penelusuran Deterministik:** Sebagai contoh ilustratif, jika pengguna bertanya *"What items are needed to make Power Treads?"*, sistem tidak sekadar menebak

berdasarkan teks, melainkan menelusuri relasi [:REQUIRES_COMPONENT] dalam graf. Penelusuran ini—baik secara langsung maupun rekursif—memungkinkan sistem memberikan jawaban yang secara faktual akurat berdasarkan topologi data yang telah didefinisikan.

III. Pembahasan

III.1. Hasil dari Sistem RAG

Berdasarkan pengujian menggunakan skrip main.py, berikut hasil *query* dari beberapa skenario uji.

Skenario Uji 1: Data Satu Hero

```
-----
Question: What is the cooldown of Anti-Mage's ultimate ability?
Generating Cypher query ....
Cypher Query:
MATCH (h:Hero {name: "Anti-Mage"})-[:HAS_ULTIMATE]->(a:Ability) RETURN a.cooldown
Executing Cypher query ....
Database Result:
{'a.cooldown': 70.0}
Generating response ....
Answer:
The cooldown of Anti-Mage's ultimate ability is 70.0.
-----
```

Analisis:

Pada skenario ini, sistem diuji untuk menjawab pertanyaan faktual spesifik mengenai properti numerik dari sebuah entitas Hero.

1. Pemahaman Entitas: Modul *Text-to-Cypher* berhasil mengidentifikasi "Anti-Mage" sebagai entitas node label :Hero.
2. Pemahaman Relasi: Sistem memahami bahwa "ultimate ability" merepresentasikan relasi [:HAS_ULTIMATE] yang menghubungkan Hero ke node :Ability.
3. Eksekusi: Kueri Cypher berhasil mengambil nilai properti cooldown dari node Ability tersebut (70.0).
4. Generasi Respons: *Response Generator* berhasil mengubah data mentah JSON {'a.cooldown': 70.0} menjadi kalimat bahasa alami yang lengkap dan akurat. Ini membuktikan bahwa graf telah berhasil mempresentasikan data yang dibutuhkan.

Skenario Uji 2: Data Beberapa Hero

```
-----
Question: Find all Carry heroes with ranged attack that has move speed bigger than 300
Generating Cypher query ....
Cypher Query:
MATCH (h:Hero)-[:HAS_ROLE]->(r:Role {name: "Carry"}),
      (h)-[:HAS_ATTACK_TYPE]->(at:AttackType {name: "Ranged"}),
      (h)-[:HAS_PRIMARY_ATTR]->(a:Attribute)
WHERE h.moveSpeed > 300
RETURN h
Executing Cypher query ....
{'h': {'attackRange': 500, 'internalId': 'npc_dota_hero_tinker', 'moveSpeed': 310, 'legs': 2, 'baseMana': 75, 'name': 'Tinker', 'baseHealth': 120}}
{'h': {'attackRange': 250, 'internalId': 'npc_dota_hero_templar_assassin', 'moveSpeed': 310, 'legs': 2, 'baseMana': 75, 'name': 'Templar Assassin', 'baseHealth': 120}}
{'h': {'attackRange': 330, 'internalId': 'npc_dota_hero_luna', 'moveSpeed': 325, 'legs': 2, 'name': 'Luna', 'baseMana': 75, 'baseHealth': 120}}
{'h': {'attackRange': 575, 'internalId': 'npc_dota_hero_leshrac', 'moveSpeed': 325, 'legs': 4, 'baseMana': 75, 'name': 'Leshrac', 'baseHealth': 120}}
{'h': {'attackRange': 365, 'internalId': 'npc_dota_hero_gyrocopter', 'moveSpeed': 320, 'legs': 2, 'baseMana': 75, 'name': 'Gyrocopter', 'baseHealth': 120}}
{'h': {'attackRange': 450, 'internalId': 'npc_dota_hero_obsidian_destroyer', 'moveSpeed': 315, 'legs': 4, 'name': 'Outworld Devourer', 'baseMana': 75, 'baseHealth': 120}}
{'h': {'attackRange': 550, 'internalId': 'npc_dota_hero_lone_druid', 'moveSpeed': 325, 'legs': 2, 'baseMana': 75, 'name': 'Lone Druid', 'baseHealth': 120}}
Generating response ....
Answer:
Here are the Carry heroes with ranged attack and move speed bigger than 300:

1. Drow Ranger – Move Speed: 310
2. Shadow Fiend – Move Speed: 305
3. Zeus – Move Speed: 315
4. Tinker – Move Speed: 310
5. Templar Assassin – Move Speed: 310
6. Luna – Move Speed: 325
7. Leshrac – Move Speed: 325
8. Gyrocopter – Move Speed: 320
9. Outworld Devourer – Move Speed: 315
10. Lone Druid – Move Speed: 325
```

Analisis:

Skenario ini menguji kemampuan sistem dalam melakukan penyaringan data kompleks yang melibatkan berbagai kondisi.

1. *Complex Query*: Sistem berhasil menerjemahkan kebutuhan pengguna ke dalam kueri Cypher yang menggunakan klausa WHERE majemuk. Kueri menggabungkan filter berdasarkan relasi [:HAS_ROLE] (untuk role "Carry"), relasi [:HAS_ATTACK_TYPE] (untuk tipe "Ranged"), dan properti numerik moveSpeed > 300.
2. *Akurasi Data*: Hasil yang dikembalikan (Drow Ranger, Shadow Fiend, Gyrocopter, dll.) sesuai dengan logika permainan Dota 2.
3. *Format Output*: LLM mampu memproses daftar hasil (*list of nodes*) yang cukup panjang dari database dan menyajikannya dalam format *bullet points* yang mudah dibaca pengguna, menunjukkan kemampuan *formatting* yang baik dari modul generasi respons.

Skenario Uji 3: Data Satu Item Sederhana


```

-----
Question: How much does a Blink Dagger cost and what does it do?
Generating Cypher query ....
Cypher Query:
MATCH (i:Item {name: "Blink Dagger"}) RETURN i.cost, i.lore, i.notes
Executing Cypher query ....
Database Result:
{'i.cost': 2250, 'i.lore': 'The fabled dagger used by the fastest assassin ever to walk the lands.', 'i.notes': "Self-casting will cause you to teleport in the direction of your team's fountain."}
Generating response ....
Answer:
The Blink Dagger costs 2250 and is described as "The fabled dagger used by the fastest assassin ever to walk the lands." It has a note that self-casting will cause you to teleport in the direction of your team's fountain.
-----

```

Analisis:

Skenario ini menguji kemampuan sistem menjawab pertanyaan majemuk yang menanyakan dua hal sekaligus: harga ("how much") dan fungsi ("what does it do").

1. Retrieval Atribut Ganda: Kueri Cypher yang dihasilkan secara efisien mengambil beberapa properti sekaligus (i.cost, i.lore, i.notes) dari node :Item dengan nama "Blink Dagger".
2. Hasil: Jawaban mencakup harga (2250) dan penjelasan fungsi (teleportasi), menjawab kedua bagian pertanyaan pengguna secara bersamaan tanpa perlu dua kueri terpisah.

Skenario Uji 4: Data Beberapa Item Sederhana

```

-----
Question: What items are required to build a Magic Wand?
Generating Cypher query ....
Cypher Query:
MATCH (i:Item {name: "Magic Wand"})-[:REQUIRES_COMPONENT]->(c:Item) RETURN c.name
Executing Cypher query ....
Database Result:
{'c.name': 'Iron Branch'}
{'c.name': 'Magic Stick'}
Generating response ....
Answer:
The items required to build a Magic Wand are Iron Branch and Magic Stick.
-----

```

Analisis:

Skenario ini berfokus pada pengujian relasi hierarkis atau komposisi antar item.

1. Penelusuran Relasi: Sistem berhasil memanfaatkan relasi semantik [:REQUIRES_COMPONENT] untuk menemukan item-item komponen dari item "induk".
2. Arah Relasi: Kueri Cypher secara tepat menggunakan arah panah relasi dari *Item Jadi* menuju *Komponen* ((i)-[:REQUIRES_COMPONENT]->(c)).
3. Hasil: Sistem secara akurat mengidentifikasi bahwa Magic Wand membutuhkan "Iron Branch" dan "Magic Stick". Ini membuktikan bahwa struktur graf efektif untuk memodelkan relasi data.

Skenario Uji 5: Data Satu Item Sederhana

```
Question: Explain about item named tango
Generating Cypher query ....
Cypher Query:
MATCH (i:Item {name: "Tango"}) RETURN i.lore, i.notes, i.attributes, i.cost
Executing Cypher query ....
Database Result:
{'i.lore': 'Forage to survive on the battlefield.', 'i.notes': '', 'i.attributes': '{"health_regen": "7.0", "buff_duration": "16.0", "tooltip_charges": "3"}', 'i.cost': 90}
Generating response ....
Answer:
The item named Tango has the following properties:

* Lore: Forage to survive on the battlefield.
* Notes:
* Attributes: It provides 7.0 health regeneration, has a buff duration of 16.0, and has 3 charges.
* Cost: 90.
```

Analisis:

Skenario ini menguji kemampuan sistem menangani permintaan yang bersifat terbuka (*open-ended*) seperti "Explain about item...".

1. Seleksi Properti: Karena pertanyaan tidak spesifik menanyakan satu atribut, LLM menghasilkan kueri yang mengambil hampir seluruh atribut penting dari item "Tango" (lore, notes, attributes, cost).
2. Penyajian Data JSON: Data atribut dalam graf disimpan sebagai string JSON. Analisis menunjukkan bahwa LLM mampu mem-*parsing* informasi yang ada di dalam struktur JSON tersebut dan menyajikannya kembali sebagai informasi yang dapat dibaca manusia. Hal ini menunjukkan representasi dari hasil database sudah dalam bentuk yang cukup jelas untuk digunakan LLM sebagai konteks.

Skenario Uji 6: Data Satu Item Kompleks

```
Question: explain in detail about divine rapier
Generating Cypher query ....
Cypher Query:
MATCH (i:Item {name: "Divine Rapier"})
OPTIONAL MATCH (i)-[:HAS_ABILITY]->(ia:ItemAbility)
OPTIONAL MATCH (i)-[:REQUIRES_COMPONENT]->(c:Item)
OPTIONAL MATCH (ia)-[:HAS_BEHAVIOR]->(b:Behavior)
RETURN i, ia, c, b
Executing Cypher query ....
Database Result:
{'i': {'notes': 'If Divine Rapier is dropped and picked up by an enemy of its original owner, it cannot be dropped again except by death.', 'cost': 5600, 'lore': 'So powerful, it cannot have a single owner.', 'name': 'Divine Rapier', 'attributes': '{"bonus_spell_amp": "25", "bonus_damage": "250", "bonus_damage_base": "100"}', 'key': 'rapier'}, 'ia': {'name': 'Everlasting', 'description': 'Dropped on death, and cannot be destroyed. \n\nBecomes unusable if picked up by an ally of its owner until it is returned to its owner. It is immediately usable by anybody if an enemy of the owner picks it up and is killed. A dropped Rapier cannot be picked up by a courier.', 'type': 'passive'}, 'c': {'cost': 3400, 'notes': '', 'lore': 'An ancient weapon that often turns the tides of war.', 'name': 'Sacred Relic', 'attributes': '{"bonus_damage": "55"}', 'key': 'relic'}, 'b': None}
{'i': {'notes': 'If Divine Rapier is dropped and picked up by an enemy of its original owner, it cannot be dropped again except by death.', 'cost': 5600, 'lore': 'So powerful, it cannot have a single owner.', 'name': 'Divine Rapier', 'attributes': '{"bonus_spell_amp": "25", "bonus_damage": "250", "bonus_damage_base": "100"}', 'key': 'rapier'}, 'ia': {'name': 'Everlasting', 'description': 'Dropped on death, and cannot be destroyed. \n\nBecomes unusable if picked up by an ally of its owner until it is returned to its owner. It is immediately usable by anybody if an enemy of the owner picks it up and is killed. A dropped Rapier cannot be picked up by a courier.', 'type': 'passive'}, 'c': {'cost': 2200, 'notes': '', 'lore': 'One of the oldest weapons forged by the Demon-Smith Abzidian, it killed its maker when he tested its edge.', 'name': 'Demon Edge', 'attributes': '{"bonus_damage": "40"}', 'key': 'demon edge'}, 'b': None}
{'i': {'notes': 'If Divine Rapier is dropped and picked up by an enemy of its original owner, it cannot be dropped again except by death.', 'cost': 5600, 'lore': 'So powerful, it cannot have a single owner.', 'name': 'Divine Rapier', 'attributes': '{"bonus_spell_amp": "25", "bonus_damage": "250", "bonus_damage_base": "100"}', 'key': 'rapier'}, 'ia': {'name': 'Transmute', 'description': 'Toggle to gain either 25% bonus spell amplification or 250 bonus attack damage.', 'type': 'toggle'}, 'c': {'cost': 3400, 'notes': '', 'lore': 'An ancient weapon that often turns the tides of war.', 'name': 'Sacred Relic', 'attributes': '{"bonus_damage": "55"}', 'key': 'relic'}, 'b': None}
{'i': {'notes': 'If Divine Rapier is dropped and picked up by an enemy of its original owner, it cannot be dropped again except by death.', 'cost': 5600, 'lore': 'So powerful, it cannot have a single owner.', 'name': 'Divine Rapier', 'attributes': '{"bonus_spell_amp": "25", "bonus_damage": "250", "bonus_damage_base": "100"}', 'key': 'rapier'}, 'ia': {'name': 'Transmute', 'description': 'Toggle to gain either 25% bonus spell amplification or 250 bonus attack damage.', 'type': 'toggle'}, 'c': {'cost': 2200, 'notes': '', 'lore': 'One of the oldest weapons forged by the Demon-Smith Abzidian, it killed its maker when he tested its edge.', 'name': 'Demon Edge', 'attributes': '{"bonus_damage": "40"}', 'key': 'demon edge'}, 'b': None}
```

```

Answer:
The Divine Rapier is a powerful item in the game with a cost of 5600 gold. It has several notable attributes, including a bonus spell amplification of 25%, a bonus damage of 250, and a bonus damage base of 100.

The Divine Rapier has two abilities: "Everlasting" and "Transmute". The "Everlasting" ability allows the rapier to be dropped on death and cannot be destroyed. If picked up by an ally of its owner, it becomes unusable until it is returned to its owner. However, if an enemy of the owner picks it up and is killed, it becomes immediately usable by anybody.

The "Transmute" ability is a toggle that allows the user to gain either 25% bonus spell amplification or 250 bonus attack damage.

The Divine Rapier requires a component item to be crafted. There are two possible component items: the Sacred Relic, which costs 3400 gold and provides a bonus damage of 55, and the Demon Edge, which costs 2200 gold and provides a bonus damage of 40.

It's worth noting that if the Divine Rapier is dropped and picked up by an enemy of its original owner, it cannot be dropped again except by death.

```

Analisis:

Skenario ini menguji entitas yang memiliki struktur lebih rumit, yaitu pertanyaan detail mengenai item yang memiliki *Ability* aktif/pasif dan *Behavior* khusus.

1. Penggunaan OPTIONAL MATCH: Sistem menghasilkan kueri menggunakan OPTIONAL MATCH. Ini sangat krusial karena tidak semua item memiliki kemampuan aktif atau *behavior* tertentu. Penggunaan ini mencegah *error* atau hasil kosong jika properti tersebut tidak ada.
2. Agregasi Informasi: Sistem berhasil menarik informasi dari node utama :Item (Divine Rapier), node terhubung :ItemAbility (kemampuan pasif), dan node :Behavior.
3. Pembuatan Jawaban: Jawaban akhir yang dibuat cukup mendetail, menjelaskan mekanik "Everlasting" (jatuh saat mati) yang merupakan fitur unik Divine Rapier, menunjukkan bahwa sistem RAG mampu menjelaskan mekanik game yang kompleks, bukan hanya data statis.

Skenario Uji 7: Penalaran dengan Satu Data Item

```

Question: What happen if i use blade mail item skill
Generating Cypher query ....
Cypher Query:
MATCH (i:Item {name: "Blade Mail"})-[:HAS_ABILITY]->(ia:ItemAbility) RETURN ia.description
Executing Cypher query ....
Database Result:
{'ia.description': 'Everytime you are attacked, you return 10 damage plus 15% of the attack damage dealt to you.'}
Generating response ....
Answer:
When you use the Blade Mail item skill, it will return 10 damage plus 15% of the attack damage dealt to you to the attacker.

```

Analisis:

Skenario ini menguji kemampuan *reasoning* sistem berdasarkan konteks, bukan sekadar pengambilan fakta.

1. Pertanyaan Hipotetis: Pengguna bertanya "What happen if..." yang memerlukan pemahaman sebab-akibat.
2. Konversi Konteks ke Logika: Kueri Cypher mengambil deskripsi kemampuan item "Blade Mail". Kunci keberhasilannya ada pada tahap *Generation*. LLM membaca deskripsi "Return damage..." dari database dan merumuskannya menjadi jawaban "When you use... it will return 10 damage plus...".

3. Kemampuan menjelaskan dari konteks: Sistem RAG tidak hanya berfungsi sebagai mesin pencari, tetapi juga sebagai asisten yang dapat menjelaskan konsekuensi dari sebuah aksi dalam game berdasarkan data teks yang tersimpan.

Skenario Uji 8: Penalaran Banyak Data Item

```
Question: Sebutkan beberapa opsi item dengan ability yang cocok untuk dipakai damage dealer
Generating Cypher query ....
Cypher Query:
MATCH (i:Item)-[:HAS_ABILITY]->(ia:ItemAbility)
RETURN i.name AS ItemName, ia.name AS AbilityName, ia.description AS AbilityDescription
Executing Cypher query ....
Database Result:
{'ItemName': 'Blink Dagger', 'AbilityName': 'Blink', 'AbilityDescription': 'Teleport to a target point up to 1200 units away. \n\nBlink Dagger cannot be used for 3 seconds after taking damage from an enemy hero or Roshan.'}
{'ItemName': 'Overwhelming Blink', 'AbilityName': 'Overwhelming Blink', 'AbilityDescription': 'Teleport to a target point up to 1200 units away.\n\nAfter teleportation, all enemies in a 800 AoE take damage equal to 100 + 50% of your strength and an additional 100% over time, and have 50% movement speed slow and 50 attack speed slow for 6 seconds. \n\nOverwhelming Blink cannot be used for 3 seconds after taking damage from an enemy hero or Roshan.'}
{'ItemName': 'Swift Blink', 'AbilityName': 'Swift Blink', 'AbilityDescription': 'Teleport to a target point up to 1200 units away.\n\nAfter teleportation, you gain 40% phased movement speed and +35 Agility for 6 seconds. \n\nSwift Blink cannot be used for 3 seconds after taking damage from an enemy hero or Roshan.'}
```

Database result masih sangat panjang

```
Generating response ....
Answer:
Berikut beberapa opsi item dengan ability yang cocok dipakai untuk damage dealer:

1. **Blink Dagger** - Teleport to a target point up to 1200 units away.
2. **Overwhelming Blink** - Teleport to a target point up to 1200 units away, dan memberikan damage pada musuh di sekitar.
3. **Monkey King Bar** - Memberikan kemampuan untuk Pierce through evasion dan memberikan bonus magical damage.
4. **Bloodthorn** - Memberikan kemampuan untuk Pierce through evasion dan memberikan bonus magical damage.
5. **Javelin** - Memberikan kemampuan untuk Pierce through evasion dan memberikan bonus magical damage.
6. **Divine Rapier** - Memberikan bonus attack damage dan spell amplification.
7. **Hand of Midas** - Memberikan bonus spell amplification dan attack damage.
8. **Khanda** - Memberikan bonus damage pada spell.
9. **Phylactery** - Memberikan bonus damage pada spell.

Namun perlu diingat bahwa pemilihan item yang tepat juga tergantung pada situasi permainan dan hero yang digunakan. Pastikan untuk memilih item yang sesuai dengan kebutuhan dan gaya permainan Anda.
```

Analisis:

Skenario ini merupakan *stress test* untuk kemampuan *Retrieval* dan *Filtering* pada sisi LLM.

1. Kueri Luas: Pertanyaan pengguna tentang "item yang cocok untuk damage dealer" bersifat subjektif dan tidak memiliki atribut langsung di database.
2. Strategi Pengambilan: Kueri Cypher yang dihasilkan (MATCH (i:Item)... RETURN ...) menarik *semua* item yang memiliki kemampuan. Hasil database sangat panjang dan mentah.
3. Peran Augmented Generation: Keunggulan sistem RAG terlihat di sini. LLM menerima daftar panjang item tersebut sebagai konteks, lalu menggunakan pengetahuan internalnya tentang semantik "damage dealer" (mencari kata kunci seperti *damage*, *crit*, *speed*, *pierce*, *teleport*) untuk memfilter dan memilih item yang relevan (seperti Daedalus, MKB, Divine Rapier) dari sekian banyak data yang ditarik.
4. Ini menunjukkan fleksibilitas sistem RAG, ketika kueri database spesifik tidak memungkinkan karena sifat pertanyaan yang subjektif, kemampuan penalaran LLM dapat digunakan untuk menyaring data mentah menjadi rekomendasi yang relevan.

III.2. Perbandingan antara Sistem RAG dan Sistem Berbasis Pengetahuan

Pada Tugas Besar I, kami mengembangkan Sistem Berbasis Pengetahuan (KBS) menggunakan Prolog yang mengandalkan penalaran deduktif berdasarkan aturan logika (*rule-based*). Pada Tugas Besar II, kami mengembangkan sistem *Retrieval-Augmented Generation* (RAG) yang mengandalkan kemampuan penalaran dan generatif menggunakan LLM dan Graf Pengetahuan. Berikut adalah perbandingan komprehensif antara kedua sistem tersebut dalam domain Dota 2:

1. Interaksi dan Fleksibilitas Input

- Sistem RAG: Menawarkan fleksibilitas tinggi. Pengguna dapat bertanya menggunakan bahasa alami (contoh: "*What items are needed to make Power Treads?*" atau "*Explain about Divine Rapier*"). Sistem mampu menangani variasi linguistik, sinonim, dan struktur kalimat yang tidak baku karena kemampuan pemrosesan bahasa alami dari LLM.
- Sistem Berbasis Pengetahuan: Bersifat kaku dan formal. Interaksi memerlukan input predikat yang spesifik dan sintaksis yang tepat (contoh: `is_glass_cannon(Hero)`. atau `is_strength_hero(X)`.). Pengguna harus memahami struktur predikat internal sistem untuk mendapatkan jawaban.

2. Metode Penalaran dan Pengambilan Informasi

- Sistem RAG: Menggunakan pendekatan *retrieval* dinamis. Sistem menerjemahkan pertanyaan ke dalam kueri basis data (Cypher) untuk mengambil fakta, lalu menggunakan LLM untuk menyusun jawaban. Sistem ini unggul dalam menjawab pertanyaan faktual, deskriptif, dan navigasi relasi (seperti resep item atau properti hero).
- Sistem Berbasis Pengetahuan: Menggunakan *logical inference*. Sistem ini unggul dalam tugas klasifikasi dan kecepatan inferensi. Contohnya, sistem dapat menyimpulkan bahwa hero adalah "Glass Cannon" bukan karena data tersebut tertulis di database, melainkan karena hero tersebut memenuhi premis logika (`Role = Nuker/Carry` DAN `Role != Durable`).

3. Deterministik vs Probabilistik

- Sistem RAG: Bersifat probabilistik. Terdapat kemungkinan LLM melakukan kesalahan dalam menerjemahkan *Natural Language* ke *Cypher* (terutama pada pertanyaan yang ambigu), atau memberikan respons yang kurang tepat jika konteks yang diambil dari Neo4j tidak lengkap.
- Sistem Berbasis Pengetahuan: Bersifat deterministik. Jika aturan dan fakta dasarnya benar, sistem akan *selalu* memberikan jawaban yang konsisten dan benar secara logika (100% akurasi berdasarkan aturan). Tidak ada risiko "halusinasi" dalam sistem ini.

4. Skalabilitas dan Pemeliharaan

- Sistem RAG: Lebih mudah diskalakan dari sisi data. Menambahkan item atau hero baru hanya perlu memasukkan node baru ke Neo4j atau menjalankan ulang *seeder* pada

data yang sudah diperbarui. LLM akan otomatis dapat menjawab pertanyaan terkait data baru tersebut.

- Sistem Berbasis Pengetahuan: Lebih kompleks dalam pemeliharaan aturan. Jika ada mekanik permainan baru (misalnya konsep *Facet* yang baru diperkenalkan), pengembang harus merancang aturan logika baru secara manual di file Prolog untuk mendefinisikan bagaimana *Facet* mempengaruhi klasifikasi hero.

Tabel Ringkasan Perbandingan:

Kriteria	Sistem Berbasis Pengetahuan	Sistem RAG
Teknologi Inti	Prolog, Ontologi (OWL)	Neo4j, LLM (Llama), Python
Input Pengguna	Predikat Logika Formal	Bahasa Alami (<i>Natural Language</i>)
Kekuatan Utama	Kebenaran yang dapat dipastikan	Pencarian Informasi & Eksplorasi Data
Output	Boolean (True/False) atau List Variabel	Teks Naratif yang Kontekstual
Penanganan Data	<i>Rule deduction</i>	<i>Retrieval</i>
Risiko Utama	Kompleksitas aturan yang sulit dikelola	Kesalahan konversi <i>Text-to-Cypher</i> , halusinasi

Secara keseluruhan, sistem Berbasis Pengetahuan sangat ideal untuk kebutuhan yang membutuhkan kepastian data, sedangkan Sistem RAG jauh lebih unggul sebagai ensiklopedia interaktif bagi pemain yang ingin mencari data spesifik tanpa harus mempelajari bahasa kueri.

III.3. Keterbatasan Graf Pengetahuan dan RAG

Meskipun sistem RAG yang dikembangkan mampu menjawab ranah pertanyaan yang lebih luas dibandingkan sistem berbasis aturan, terdapat beberapa **keterbatasan dari sistem yang telah dikembangkan** yang ditemukan selama pengembangan dan pengujian:

1. Keterbatasan Cakupan Data Dinamis

Graf pengetahuan yang dibangun berdasarkan data JSON (heroes.json, items.json, dll.) bersifat **statis**. Sistem hanya merepresentasikan data "snapshot" dari permainan (seperti *base damage*, *cooldown*, dan biaya *mana*).

- Implikasi: Sistem tidak dapat menjawab pertanyaan yang melibatkan kalkulasi dinamis atau simulasi keadaan permainan, seperti "*Berapa physical damage reduction yang dimiliki Axe level 25 dengan item Shiva's Guard?*". Graf tidak menyimpan logika perhitungan *scaling* atribut per level atau interaksi penumpukan antar item.

```
-----
Question: Berapa physical damage reduction yang dimiliki Axe level 25 dengan item Shiva's Guard?
Generating Cypher query ....
Cypher Query:
Question is irrelevant to the database context.
```

2. Isu Keterbatasan pada Terminologi Umum

Sistem RAG sangat bergantung pada kemampuan LLM atau skrip untuk memetakan istilah input pengguna ke properti name yang ada di database Neo4j.

- Implikasi: Jika pengguna menggunakan istilah *slang* atau singkatan komunitas yang tidak ada dalam atribut dname atau localized_name (misalnya menyebut "*Sheepstick*" untuk "*Scythe of Vyse*"), kueri Cypher yang dihasilkan mungkin gagal menemukan *node* yang relevan (RETURN NULL), kecuali LLM memiliki pengetahuan internal mengenai sinonim tersebut.

```
Question: Tell me about Sheepstick
Generating Cypher query ....
Cypher Query:
MATCH (i:Item {name: "Sheepstick"}) RETURN i
Executing Cypher query ....
Database Result:
(no result)
Generating response ....
Answer:
There is no item named "Sheepstick" in the database.
```

3. Ketergantungan pada Akurasi *Text-to-Cypher*

Kualitas jawaban sangat bergantung pada validitas sintaksis dan semantik dari kueri Cypher yang dihasilkan oleh LLM pada tahap pertama.

- Implikasi: Untuk pertanyaan yang kompleks atau ambigu, LLM berpotensi menghasilkan kueri yang salah secara logika atau tidak menghasilkan query sama sekali. Jika kueri Cypher salah, hasil *retrieval* akan kosong atau tidak relevan, sehingga menyebabkan jawaban akhir menjadi salah.

```
Question: Hero mana yang memiliki effective HP tertinggi terhadap magic damage?
Generating Cypher query ....
Cypher Query:
Question is irrelevant to the database context.
```

4. Nuance Mekanik Permainan yang Hilang

Skema graf saat ini (schema_example.txt) menyederhanakan beberapa aspek mekanik yang sangat mendetail.

- **Implikasi:** Atribut seperti *bkb Pierce* (apakah skill menembus kekebalan sihir) atau tipe *dispel* (Basic/Strong) ada di data mentah JSON, namun jika skema graf tidak secara eksplisit mendefinisikan properti ini atau jika LLM tidak dilatih untuk memperhatikannya, sistem mungkin gagal menjawab pertanyaan nuansa seperti "*Does Primal Roar pierce spell immunity?*" dengan akurat.

```
Question: Does Primal Roar pierce spell immunity?
Generating Cypher query ....
Cypher Query:
MATCH (a:Ability {name: "Primal Roar"})-[:HAS_BEHAVIOR]->(b:Behavior) RETURN b.name
Executing Cypher query ....
Database Result:
{'b.name': 'Unit Target'}
Generating response ....
Answer:
No, Primal Roar does not pierce spell immunity because it has the "Unit Target" behavior, which is generally affected by spell immunity.
```

IV. Kesimpulan

Pengembangan sistem *Retrieval-Augmented Generation* (RAG) berbasis Graf Pengetahuan untuk domain Dota 2 dalam tugas besar ini telah berhasil diimplementasikan dan menunjukkan hasil yang signifikan dalam menjembatani interaksi bahasa alami dengan data permainan yang kompleks. Berdasarkan tahapan pengembangan dan pengujian yang telah dilakukan, dapat ditarik beberapa kesimpulan utama:

1. **Efektivitas Arsitektur Graph RAG** Sistem yang dibangun menggunakan pendekatan *Text-to-Cypher* terbukti efektif dalam menangani kueri yang membutuhkan pemahaman relasional mendalam. Berbeda dengan RAG konvensional yang berbasis vektor teks, integrasi Neo4j sebagai *ground truth* memungkinkan sistem menjawab pertanyaan faktual (seperti spesifikasi item) dan pertanyaan navigasi (seperti pohon resep atau hierarki skill) dengan tingkat determinisme yang tinggi.
2. **Kemampuan Penalaran dan Fleksibilitas** Pengujian terhadap berbagai skenario menunjukkan bahwa kombinasi LLM (Llama 4 Scout) dan struktur graf mampu menangani variasi pertanyaan yang luas, mulai dari pengambilan data atribut tunggal hingga pertanyaan majemuk yang melibatkan filter kondisi bertingkat. Sistem juga menunjukkan kemampuan "penalaran" terbatas dengan mensintesis data mentah

menjadi penjelasan naratif yang koheren, seperti menjelaskan implikasi penggunaan *skill* item tertentu.

3. **Perbandingan dengan Sistem Berbasis Pengetahuan (Rule-Based)** Dibandingkan dengan sistem berbasis logika (Prolog) yang dikembangkan sebelumnya, sistem RAG ini menawarkan keunggulan signifikan dalam hal **fleksibilitas interaksi**. Pengguna dapat mengeksplorasi data menggunakan bahasa alami tanpa perlu memahami sintaks predikat formal. Namun, fleksibilitas ini hadir dengan *trade-off* berupa sifat probabilistik, di mana akurasi jawaban sangat bergantung pada ketepatan penerjemahan niat pengguna menjadi kueri Cypher oleh LLM.
4. **Keterbatasan Sistem** Meskipun fungsional, sistem ini memiliki batasan dalam menangani **data dinamis** dan **nuansa mekanik**. Karena graf bersifat statis ("snapshot"), sistem tidak dapat melakukan simulasi perhitungan *real-time* (seperti reduksi *damage* berdasarkan level). Selain itu, ketergantungan pada skema yang disederhanakan menyebabkan hilangnya beberapa detail mekanik mikro (seperti interaksi *spell immunity* yang spesifik) dan kesulitan dalam mengenali istilah *slang* komunitas yang tidak terpetakan dalam database.

Secara keseluruhan, sistem ini berhasil membuktikan bahwa penggunaan Graf Pengetahuan sebagai basis konteks bagi LLM adalah solusi yang lebih superior dibandingkan metode pencarian teks biasa untuk domain yang sangat terstruktur seperti video game. Sistem ini berfungsi ideal sebagai ensiklopedia interaktif yang cerdas, meskipun masih memerlukan pengembangan lebih lanjut pada sisi ontologi dan pemrosesan istilah untuk mencapai tingkat akurasi setara ahli manusia.

Lampiran

Matriks Kontribusi

No.	Nama Mahasiswa	NIM	Deskripsi Kontribusi
1.	Sa'ad Abdul Hakim	13522092	Laporan, pembuatan knowledge graf
2.	Rayhan Fadhlan Azka	13522095	Laporan, insiasi awal knowledge graf dan RAG
3.	Rayendra Althaf Taraka Noor	13522107	Laporan, Detail Item

Link Repository Github

<https://github.com/RayhanFadhlan/rpp-tubes2>